Universidade Federal da Paraíba
Centro de Ciências Aplicadas e Educação
Departamento de Ciências Exatas
Bacharelado em Sistemas de Informação
Licenciatura em Ciências da Computação

Relatório do projeto Banco Imobiliário (Etapa 3 - Stories 8 e 9)

Disciplina: Programação Orientada a Objetos
Equipe: Equipe 2
Link Github: dcx-cursos/projeto-poo-2019-1-jo/tree/dev
Membros: Amanda Azevedo Martins
Clebson Augusto Fonseca
Joana Darck Soares da Silva
Joyce Sousa dos Santos

1. Introdução

1.1. Objetivos

Banco imobiliário é uma versão de um dos jogos de tabuleiro mais conhecidos e populares do mundo, o Monopoly. Neste projeto foi implementada uma versão em Java, que funciona com uma interface textual simples e opera com base nas regras brasileiras do jogo. O projeto Banco Imobiliário foi desenvolvido pelo aluno, do terceiro período de licenciatura em Ciências da Computação, Clebson Augusto Fonseca e pelas alunas, também do terceiro período, de bacharelado em Sistemas de Informação, Amanda Azevedo Martins, Joana Darck Soares da Silva e Joyce Sousa dos Santos.

1.2. Tecnologias

As tecnologias usadas para a conclusão desta segunda entrega foi, a linguagem de programação Java8, a IDE (Integrated Development Environment) "Eclipse Java 2019-06" para desenvolvimento do projeto. Utilizamos o GitHub, para armazenamento e versionamento do código. Além disso utilizamos a framework de testes, JUnit4 o Mockito com a finalidade simulação de algumas classes, para que podermos testar algumas funcionalidades e para criação do diagrama de classes em UML (Unified Modeling Language), utilizamos o "Astah UML".

1.3. Entregas e Datas

As datas de entrega deste projeto está seguindo o calendário estabelecido pelo professor Fábio, sendo assim, conseguimos entregar a primeira versão dia 12 de agosto de 2019, a segunda versão no dia 02 de setembro de 2019 e a terceira no dia 22 de setembro de 2019.

1.4. Ações desenvolvidas por cada participante

Nome	Descrição
Amanda Azevedo Martins	Parte do Story 8 e parte do JavaDoc.
Clebson Augusto Fonseca	Parte do Storie 9, mudança de factory method para o template method e padronização das entradas.

Joana Darck Soares da Silva	Parte do Story 9 e todos os testes.
Joyce Sousa dos Santos	Implementação do padrão Singleton, parte do JavaDoc e parte do Story 8.

2. Metodologia

Primeiramente, foram feitas as alterações de correção da segunda entrega, em seguida desenvolvemos o storie 8, Construção de casas e o storie 9, Venda de casas. Simultâneo a isso, foram desenvolvidos os testes do código que estava sendo implementado. Posteriormente, documentamos o código, aprimoramos a terceira versão da UML e redigimos o relatório.

3. Instalação e execução

3.1. Primeiramente, certifique-se de que você tenha instalados:

- JDK 8
- Eclipse ou Netbeans
- git

3.2. Em seguida crie um diretório para ser sua Workspace e clone o repositório dentro desta pasta:

```
git clone https://github.com/dcx-cursos/projeto-poo-2019-1-jo.git
```

3.3. Após ter realizado o passo acima, mude a branch para "entrega3"

```
git checkout entrega3
```

3.4. Agora abra o projeto na sua IDE selecionando a opção:

Para o Eclipse	> File > Open Projects from File System...
Para o NetBeans	> Arquivo > Abrir Projeto

3.5. Para jogar Banco Imobiliário execute a classe Main.java, do pacote src/ufpb/jogo.

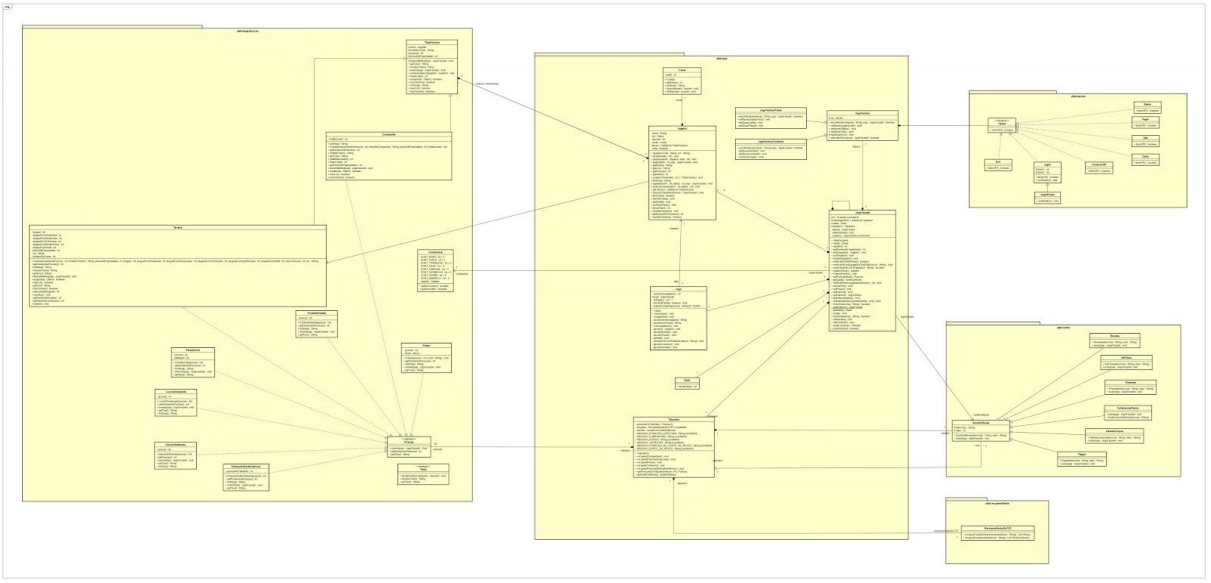
3.6. Para verificar os testes feitos execute a classe AllTests.java do pacote src/ufpb/tests.

- Para verificar os testes é importante que as seguintes dependências estejam devidamente configuradas:
  - JUnit4
  - Mockito

4. Desenvolvimento

4.1. Diagrama de classes

4.1.1. Diagrama de classes UML



4.1.2. Descrição do diagrama de classes UML

Nome do Pacote	Descrição
ufpb.jogo	Tabuleiro.java se relaciona com SorteOuReves.java, com recuperaDadosDoTXT.java, com Posicao.java […] Jogador.java se relaciona com TituloFactory,java […] JogoFactory.java com Opcao.java,JogoFactoryPrisao.java[…]JogoFacade.java

ufpb.opcoes	Este pacote é referente às opções que cada jogador tem durante sua jogada. Sendo elas, a opção de jogar - jogar os dados e se movimentar no tabuleiro, de ver o status do jogador(se ele tem títulos, mostrar quais são; o saldo do jogador e a posição no qual está situado), de tentar usar a carta de Habeas Corpus para sair da prisão, de pagar para sair da prisão, de jogar os dados para tentar sair da prisão e a opção de sair do jogo. No padrão JogoFactory, são as opções presentes na “fábrica”, o JogoFactory cria tipos de opções.
ufpb.cartas	Neste pacote são armazenadas as classes que extends a superclasse, SorteOuReves.java. Nela foi criada uma classe para cada tipo de carta de Sorte ou Revés, que realizam ações com base na sua descrição. Na classe de SorteOuReves.java, onseguimos agrupar as cartas como sendo do tipo Pague, em que o jogador tem que pagar algum valor ao banco, Presente, em que todos os jogadores pagam alguma quantia ao jogador que retirou esse carta da pilha, Receba, em que o jogador recebe uma quantia do banco, Vá para prisão, em que o jogador é mandado para a prisão, Habeas Corpus, em que o jogador ao possuir esta carta pode-se livrar da prisão e Sorte ou Revés, que a depender do resultado da soma dos dados, ou ele paga ou recebe alguma quantia do banco. Neste pacote são armazenadas as classes que extends a superclasse, Sorte ou revés. Nela foi criada uma classe para cada tipo descrito acima, que realizam ações com base na sua descrição.
ufpb.lougradouros	Por implementar Posicao.java, todas as classes deste pacote possuem um método evento(), esse método

	possibilita que ao o jogador se mover no tabuleiro e parar em uma nova posição, ele executa apenas a ação prevista para aquela posição.
ufpb.exception	Todas as exceções deste pacote são lançadas e tratadas nas classes Conta.java, Jogador.java, JogoFacade.java, JogoFactory.java, JogoFactoryPrisao.java e Jogo.java, do pacote ufpb.jogo , tais como, se a cor que um jogador escolheu para si é uma cor possível, se um valor de dinheiro é válido , se algum limite foi excedido, entre outros. Este pacote possibilita que existam exceções mais especializadas, para que possamos tratá las para entender melhor alguns erros.
ufpb.recuperaDados	Este pacote é referente a camada de persistência de dados em arquivos, onde conseguimos recuperar os dados, em forma de String dos arquivos .txt, para serem convertidos em objetos na classe Tabuleiro.java.

4.2. Padrões de Projeto utilizados

Facade	Foi utilizado como uma fachada de comunicação da lógica do banco imobiliário com a parte visível para o jogador.
Factory	Foi utilizado para tratar as opções dos jogadores, quando se está na prisão, uma série de opções está disponível e quando não está as opções normais são usadas. Também foi usado na implementação de títulos que tinham um evento muito parecido.
Strategy	Foi utilizado para que em tempo de execução fosse feito um evento/ação do baralho/tabuleiro.

Singleton	Foi utilizado para ter uma classe JogoFacade com uma única instância.